The purpose of this project was to predict daily weather metrics for a given zip code based on historical weather data scraped from the web. <http://www.wunderground.com> was used to gather the historical data. The data provided by the website has readings taken roughly every hour for the following variables: Date, Time, Temperature, Dew Point, Humidity, Sea Level Pressure, Visibility, Wind Direction, Wind Speed, Gust Speed, Precipitation, Events, and Conditions. Data from 10/31/2010 to 10/31/2016 was gathered for my zip code (60502), and a zip code to the southwest (61350), as weather generally tends to travel Northeast in the northern hemisphere. The predictions of the target variables were made for my zip code, but both its historical data and the data from the zip code to the southwest were used as input variables to the models. Below is a sample of what the raw data look like:

**Figure** **1**: Snippet of raw weather data for zip code 60502 on 12/8/2016

To avoid high variance, only variables believed to be good predictors of the target variables were kept. Daily weighted averages (based on time) were then calculated for each variable. Lastly, values for yesterday’s weather metrics and average values for the previous week’s weather metrics were calculated for both zip codes. The conditions column had to be codified, i.e. assigned a numeric variable in place of the character variables when being used for prediction. The idea here was that one day alone (yesterday) couldn’t provide enough prediction value, given that weather is rather volatile in nature. To counter this, several days of weather leading up to the target date (today) were included. Once the preprocessing was complete, the final dataset had the following columns, where “Target” denotes variables being predicted and “Feature” denotes predictors of the target variables:

|  |  |
| --- | --- |
| **Variable Name** | **Variable Type** |
| 60502\_TemperatureF\_today | Target |
| 60502\_PrecipitationIn\_today | Target |
| 60502\_Conditions\_today | Target |
| 60502\_TemperatureF\_yesterday | Feature |
| 60502\_PrecipitationIn\_yesterday | Feature |
| 60502\_Conditions\_yesterday | Feature |
| 60502\_Dew.PointF\_yesterday | Feature |
| 60502\_Sea.Level.PressureIn\_yesterday | Feature |
| 60502\_Wind.SpeedMPH\_yesterday | Feature |
| 60502\_Humidity\_yesterday | Feature |
| 60502\_TemperatureF\_previous\_week | Feature |
| 60502\_PrecipitationIn\_ previous\_week | Feature |
| 60502\_Conditions\_ previous\_week | Feature |
| 60502\_Dew.PointF\_ previous\_week | Feature |
| 60502\_Sea.Level.PressureIn\_ previous\_week | Feature |
| 60502\_Wind.SpeedMPH\_ previous\_week | Feature |
| 60502\_Humidity\_previous\_week | Feature |
| 61350\_TemperatureF\_yesterday | Feature |
| 61350\_PrecipitationIn\_yesterday | Feature |
| 61350\_Conditions\_yesterday | Feature |
| 61350\_Dew.PointF\_yesterday | Feature |
| 61350\_Sea.Level.PressureIn\_yesterday | Feature |
| 61350\_Wind.SpeedMPH\_yesterday | Feature |
| 61350\_Humidity\_yesterday | Feature |
| 61350\_TemperatureF\_previous\_week | Feature |
| 61350\_PrecipitationIn\_ previous\_week | Feature |
| 61350\_Conditions\_ previous\_week | Feature |
| 61350\_Dew.PointF\_ previous\_week | Feature |
| 61350\_Sea.Level.PressureIn\_ previous\_week | Feature |
| 61350\_Wind.SpeedMPH\_ previous\_week | Feature |
| 61350\_Humidity\_previous\_week | Feature |
| Month | Feature |

**Figure 2**: List of columns in dataset

There were a few different prediction tasks being done. Today’s temperature and today’s precipitation are both continuous variables, whereas conditions is a discrete variables with values such as “Clear”, “Partly Cloudy”, “Rain”, etc. I thought it would be interesting to build models for these variables because they provide a wide variety of prediction tasks to perform. Predicting temperature is a rather traditional regression model, where the range of outputs varies a fair amount. Predicting precipitation could be a regression model, as the outcome is continuous, but the majority of days in the zip code used in this analysis have values of 0 inches of precipitation for the day. With that in mind, precipitation outcomes was changed to a Boolean variable, being assigned a value of 1 if precipitation was present for a day and a value of 0 if it was not present, making it a classification problem. Lastly, predicting conditions was a rather typical classification problem using supervised learning, where 10 different outcomes were possible. The outcomes included unknown, clear, partly cloudy, mostly cloudy, fog, light rain/drizzle, rainy, thunderstorms/heavy rain, snow, and freezing rain/sleet. The below paragraphs detail the methods, results, and interpretations of each prediction task. In all cases, 70% of the data was used in the training set and the remainder was used in the test set. With more time, validation sets would have been created to tune these models as well.

***Task 1: Predicting Temperature***

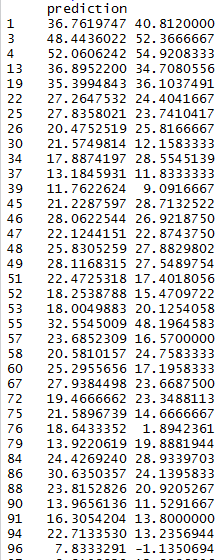
Three different regression models were used to predict temperature. These models included support vector machine, logistic regression, and random forest. The below table displays some evaluation metrics describing how the models performed on the test set:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | RMSE | Rsquared | Median Absolute Deviation |
| Generalized Linear Model | 5.2585 | 0.9336 | 4.6024 |
| Support Vector Machine | 5.2553 | 0.9336 | 4.5301 |
| Random Forest | 5.5234 | 0.9266 | 4.7982 |

**Figure 3**: Evaluation metrics of temperature prediction regression models

It’s interesting to note that the models all performed very similarly and all reasonably well. Really, any one of these models could be used, though the support vector machine was technically the best based on RMSE, Rsquared, and MAD. Rsquared of roughly 0.93 is a decent fit, but remembering the problem at hand (predicting temperature), it makes sense that the fit would be this high. The temperatures do behave similarly from year to year. RMSE of roughly 5 degrees Fahrenheit is quite reasonable, and I would submit that any of these models is a good performer when predicting temperature. It is important to keep in mind the amount of data that was available for these prediction tasks, though, especially the previous day’s weather. Most people want predictions for a week out, so in many cases, this wealth of data may not be available. I speculate that all of these algorithms performed similarly because the nature of the way they perform is not all that different. I am a little surprised the random forest regression model didn’t deviate much from the other models, but it may once again be due to the fact that the behavior of the target variable is relatively consistent.

Below is a sample of the output results when using the Support Vector Machine algorithm:



**Figure 4**: Example results of Support Vector Machine algorithm when predicting temperature (Fahrenheit)

***Task 2: Predicting Precipitation***

Three different classification models were used to predict the presence of precipitation. These models included k-nearest neighbors, Naïve Bayes, and a random forest. The below table displays some evaluation metrics describing how the models performed on the test set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **AUC** | **Sensitivity** | **Specificity** |
| Random Forest | 0.6922 | 0.5728 | 0.9029 | 0.2476 |
| Naïve Bayes | 0.6325 | 0.6076 | 0.6772 | 0.5381 |
| k-Nearest Neighbors | 0.6738 | 0.5617 | 0.8758 | 0.2476 |

**Figure 5**: Evaluation metrics of precipitation prediction classification models

Below are the confusion matrices for the various classification models:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actuals | |
| Predictions |  | **0** | **1** |
| **0** | 398 | 156 |
| **1** | 45 | 54 |

**Figure 6**: Confusion matrix for random forest classifier

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actuals | |
| Predictions |  | **0** | **1** |
| **0** | 300 | 97 |
| **1** | 143 | 113 |

**Figure 7**: Confusion matrix for Naïve Bayes classifier

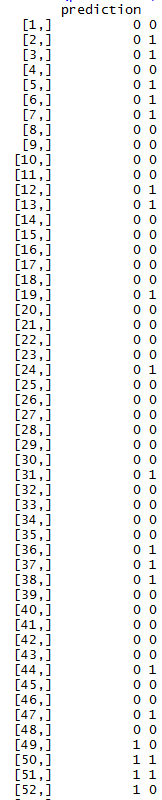
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actuals | |
| Predictions |  | **0** | **1** |
| **0** | 388 | 158 |
| **1** | 55 | 52 |

**Figure 8:** Confusion matrix for k-nearest neighbors classifier

The results of these various models was particularly interesting. The random forest had a high sensitivity, meaning it was likely to predict rain when rain actually occurred, but was also low on specificity, meaning it was likely to predict rain on days when rain didn’t end up occurring. This model would be the best to choose if one would like to ensure they are always prepared for rain. K-nearest neighbors behaved similarly, erring higher on sensitivity than specificity. Naïve Bayes was moderate in both sensitivity and specificity, though it did have the lowest accuracy. I would propose this to be the model of the 3 to use when predicting rain, as it is more moderate and a false positive or false negative isn’t typically devastating in this context.

It’s not all that surprising that the random forest and k-nearest neighbors tended to have a higher false positive rate. I could imagine that there were nodes chosen in the random forest that caused a true prediction every time, whereas Naïve Bayes tended to be less discrete with these variables.

Below is a sample of the output results when using the Naïve Bayes algorithm:



**Figure 8**: Example results of Naïve Bayes algorithm when predicting precipitation present (Boolean)

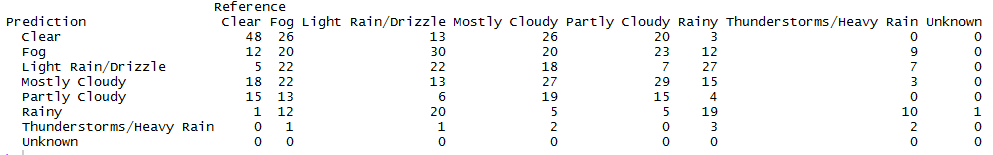
***Task 3: Predicting Conditions***

Three different classification models were used to predict the day’s conditions. These models included k-nearest neighbors, Naïve Bayes, and a random forest. The below table displays some evaluation metrics describing how the models performed on the test set:

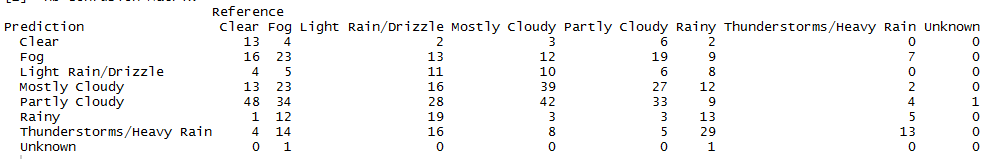
|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy** | **AUC** |
| Random Forest | 0.2442 | 0.8939 |
| Naïve Bayes | 0.2227 | 0.2929 |
| k-Nearest Neighbors | 0.1997 | 0.8788 |

**Figure 9**: Evaluation metrics of conditions prediction classification models

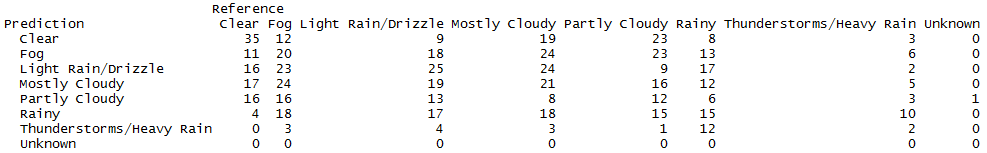
Below are the confusion matrices for the various classification models:



**Figure 10**: Confusion matrix for random forest classifier



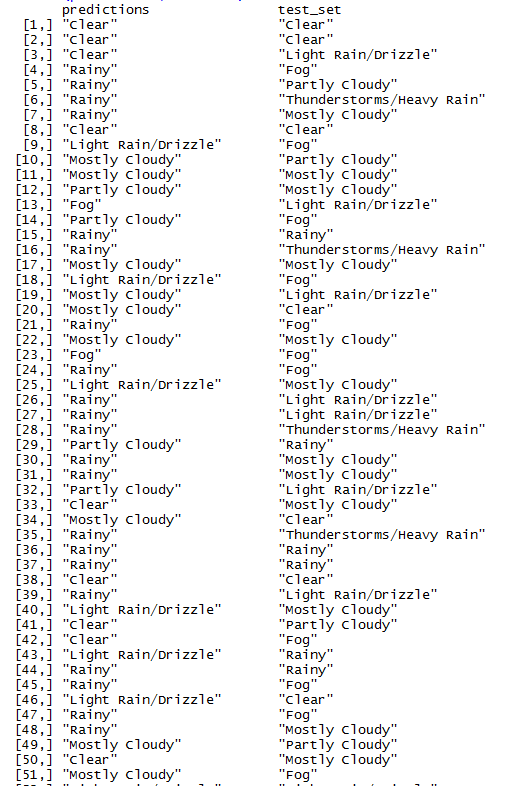
**Figure 11**: Confusion matrix for Naïve Bayes classifier



**Figure 12**: Confusion matrix for k-nearest neighbors classifier

All of these models had very low accuracy, which isn’t a huge surprise because there are so many potential classes to predict. However, AUC was rather high for both random forest and k-nearest neighbors. I would be comfortable proposing using the random forest to predict outcome, given that a miss on the classification isn’t that devastating if it predicted a class nearby, i.e. predicted “light rain” and it ended up being “rainy”. Naïve Bayes was a very poor performer.

Below is a sample of the output results when using the Random Forest algorithm:



**Figure 13**: Example results of Random Forest algorithm when predicting conditions

Overall, while these models are far from perfect, it proved to be an interesting exercise in solving a variety of prediction problems. This really illustrated the necessity of trying several different models, as the performance can widely vary among them. As mentioned earlier, it would be more ideal to try even more models, spend time tuning the models, and even spend more time working on variable selection and bringing in more variables.